Middleware as a Mission Enabler for Mars Exploration

Norman.Lamarra@jpl.nasa.gov, M/S 126-201, 818-393-1561
Anthony.Barrett@jpl.nasa.gov, M/S 126-347, 818-393-5372
Thomas.McVittie@jpl.nasa.gov, M/S 126-255, 818-393-5052
Larry.Bergman@jpl.nasa.gov, M/S 126-254, 818-393-5314
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr, Pasadena, CA 91109

Abstract-- Effective middleware can improve the capability of business and science applications in several ways, e.g., by hiding platform heterogeneity or by providing standard shared services which reduce the complexity or increase the capability of every application. Recent successes in middleware, such as multi-tier client/server and web-based architectures, have fueled phenomenal growth in "enterprise-level" applications, which provide better integration and more rapid adaptability of business in many fields. The term "web year" has been coined to describe the pace of such application evolution, in contrast to the relatively slow and costly approaches dictated by previous software generations. The "holy grail" of such application development is the ideal of platform and location independence, as exemplified by the remarkable pace of advance and adoption of Java technology in less than a decade. Unfortunately, the science and engineering community has been slower to adapt its software processes to the pace of business evolution, especially in aerospace and defense systems, partly due to the complexity, criticality, and (many-year) length of the system lifecycle for such systems. We suggest that remote planetary exploration (e.g., the Mars Program) could benefit significantly from such middleware developments, hence enabling more powerful mission applications with reduced complexity. Applicable middleware services such as navigation, data management and evolvable communications could be provided by orbiting Mars satellites, and could be used by subsequent missions (e.g., in situ) much beyond the duration of the orbiters' original primary mission. This middleware service approach provides many potential benefits, because subsequent missions can leverage capabilities already available, hence enabling cost and risk reduction via simplification. Moreover, system-wide benefits (such as fault and resource management) can also be provided via middleware. This paper will describe a study conducted at JPL on middleware implementation strategies and potential applications for deep-space systems, such as flexible communication relay stations and spacecraft constellations, paving the way for progressively effective remote exploration.

1. INTRODUCTION

NASA has planned an ongoing campaign of missions to Earth's closest neighbor planet Mars, which was first visited in 1964 by the Mariner 4 fly-by mission. Today, we are receiving telemetry from Mars Global Surveyor, and future missions are planned over the next several years. Figure 1 depicts a view of the "problem space" for communication and control of spacecraft and instruments in deep space, via JPL's Deep Space Network (DSN), showing ground processing, transmission/reception, on-board processing, and eventual science production. International bodies such as the CCSDS attempt to standardize such communications, and the newly-formed Space Standards Working Group of the Object Management Group (OMG), is also addressing software infrastructure issues. This paper addresses how software technology can assist such a Mars campaign, and suggests areas for potentially fruitful future study.

2. IMPROVING ACCESS TO MARS

Over the past forty years fifteen missions have successfully sent spacecraft to Mars. While Mariner 4 was the first spacecraft and simply flew by Mars in 1964, later spacecraft were progressively more complex and capable. Orbiters (like Mariner 9 and Mars 3) followed fly-bys in 1971. While Mars 3 also dropped a lander, it stopped transmitting twenty seconds after landing. Viking 1 heralded the age of landers by reaching the surface in 1976 and surviving for six years. Finally, Pathfinder started the age of robotic rovers in 1997. While the later missions were more complex, all missions shared a

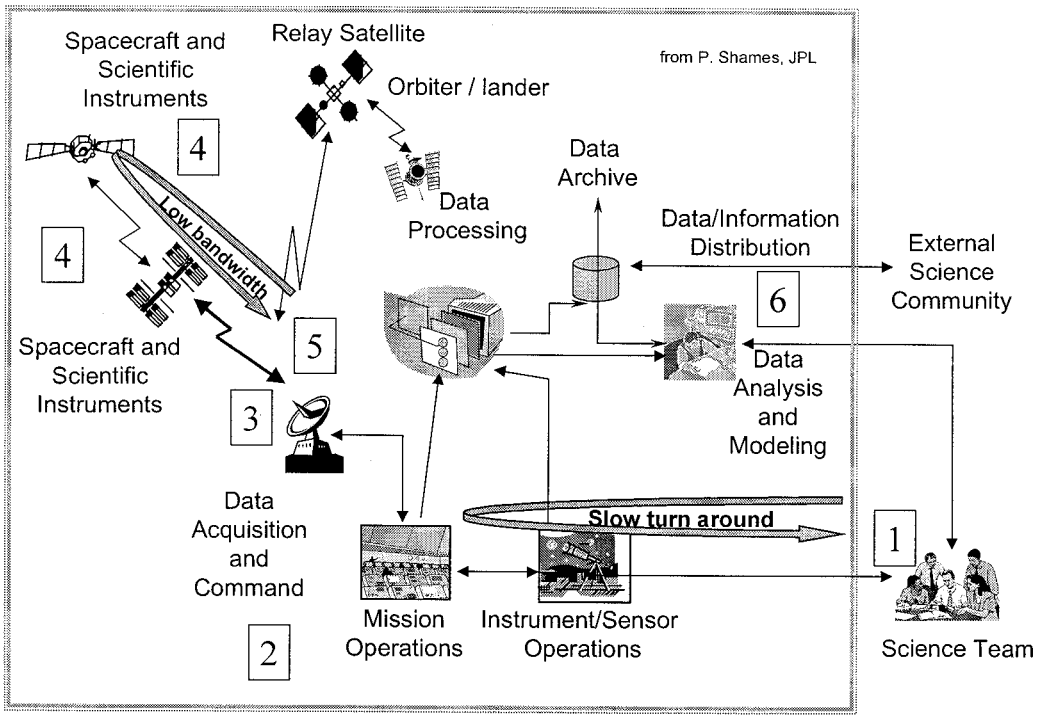

Figure 1. Problem Space Domain for Space Standards Working Group of OMG (see text)

simplifying assumption – each operated in isolation; in the future, cooperative missions will succeed such isolated missions. The international science community is planning sixteen missions to Mars over the next ten years (Figure 2), and these missions will cooperate in multiple ways. Earlier missions will provide precision approach navigation for later missions, and real-time tracking for critical events like descent and landing or orbit insertion. Orbiters will provide relay services to landed assets and positioning services to rovers and other mobile "scout" missions. All missions will cooperate on radiometric experiments and maintaining a common time reference for relating data between missions. These features are conceptualized as a "Mars Network" of orbiting satellites [Cesarone et al., 1999]. While all of the missions will improve the potential for collecting data on Mars by virtue of placing multiple sensors around Mars, actually realizing this potential requires improving mission operations tools and techniques both to command the sensors and to collect the resultant raw measurements.

Scientist to Instrument Connectivity

Within the Mars exploration context, one useful operations improvement is to make it easier for scientists to control their experiments. As depicted in Figure 1, the operations process for a spacecraft in deep space is a six-step cycle including: (1) science plan generation, (2) command sequence generation and validation, (3) uplink of the new sequence, (4) sensor data acquisition, (5) telemetry downlink, (6) science and engineering analysis,. During telemetry downlink, a spacecraft transmits mainly raw sensor data to the ground. Engineering analysis extracts spacecraft health and status, while science analysis extracts data to compute science products that help researchers answer the questions that originally motivated the mission. Often these products raise more questions than they answer, and schedules for requesting new measurements are built into the science plan generation process. These schedules are then passed to the command sequence generation and validation process where they are expanded into a sequence of commands to articulate the instruments and collect data for downlink to Earth, while avoiding any flight rule violations that could endanger the spacecraft. After uplinking the validated sequence, the cycle returns the next batch of downlinked telemetry.

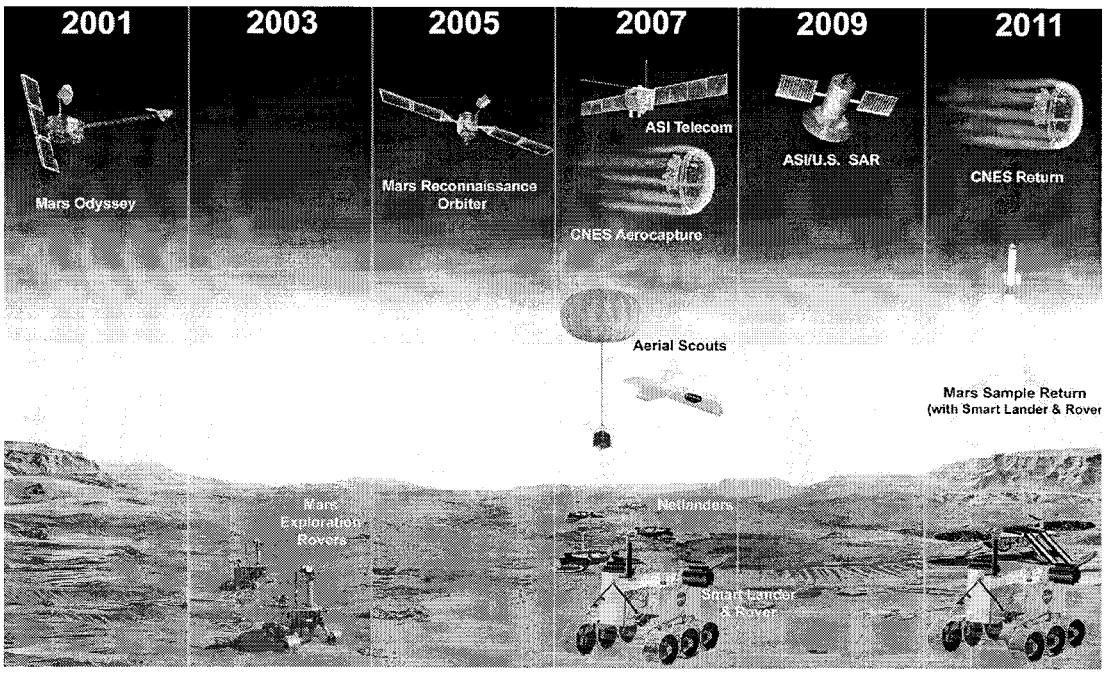

Figure 2. Future Missions to Mars

During Sojourner operations [Mishkin et al., 1998], new command sequences were uplinked early to mid-morning on Mars and data was downlinked at three times during the Martian day: just prior to the uplink, at Martian noon, and mid- to late-afternoon on Mars. While the early downlink was used to assure that no contingencies invalidated the next command sequence, the operations cycle for generating the next day's sequence started after the late-afternoon downlink. Given that Sojourner command sequences were manually generated, the daily operations efforts were arduous. Scientists had to quickly generate an initial observation plan and negotiate with the command sequencing personnel to generate a sequence to uplink. This negotiation can both add and remove observations as opportunities and problems are found while validating the sequence, but the cycle must complete before the next morning's uplink.

One way to accelerate this process and increase the amount of resultant science data uses automation both to interpret the downlinked data and to automate the command sequencing and validation steps. In this way a scientist has more time to generate an observation plan and she can continually validate observation plans during the generation process. Figure 3 illustrates such a system [Sherwood et al., 2000] that was prototyped using the ASPEN planner/scheduler and the Web Interface for TeleScience (WITS). While the ASPEN system automated much of the command sequence generation and validation process, WITS provided an interface for developing science plans, resulting in a more direct link between the scientist and her experiment.

Improved Average Downlinked Data Quality

Another approach to improving Mars access involves increasing the amount of *information* delivered to a scientist. While an instrument can produce prodigious amounts of raw data, the communications system can only deliver a small percentage of it to a mission scientist. For instance, there are plans to put a camera on the Mars Reconnaissance Orbiter that can image spots on the Martian surface to a 20cm resolution. At this resolution, there are 3.6×10^{15} pixels on the surface of Mars. Supposing 12 bits of information per pixel, and a 2:1 lossless compression algorithm, the camera can generate 2.2×10^{16} bits of information. Using the DSN, an orbiter like Mars Global Surveyor can downlink

about 85 Kbps during about 8 track hours per day, resulting in 8.5×10^{11} downlinked bits per Earth year. Thus only 0.0039% of the Martian surface can be delivered per Earth year as raw imagery.

Given that such a small fraction of the data can be downlinked, intelligent remote data handling is needed to improve the proportion of desired information in the downlinked data. Such intelligence might include techniques for remote science analysis and data mining, data compression and fusion, and improved (e.g., event-driven) data handling [Manduchi, et al., 2000]. From the perspective of Figure 3, we can implement the enhancement by migrating parts of the science-processing task onto

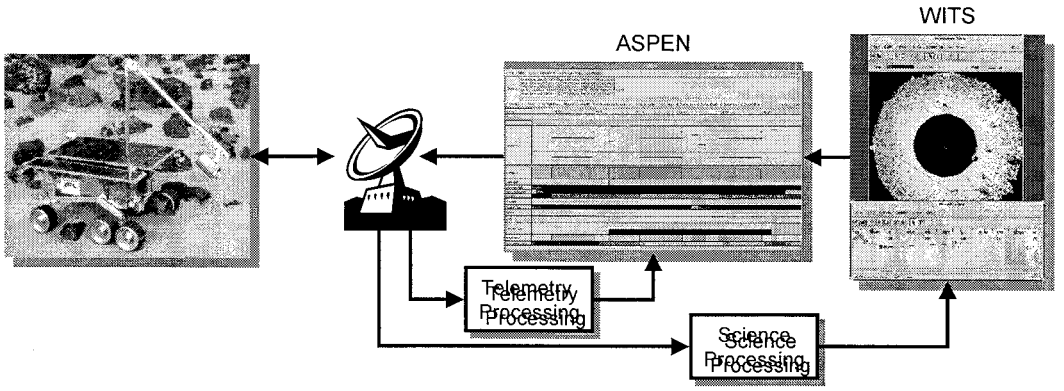

Figure 3. Automated rover command generation with ASPEN and WITS

the spacecraft in order to prioritize the raw data by its information content. For instance, a scientist might be particularly interested in smooth round Martian boulders and their surrounding environment to prove the persistent existence of running water over Martian centuries. With this in mind, the scientist could utilize an on-board pattern matching algorithm for finding pixel patterns that denote smooth round rocks. She can then request many more images, and use the algorithm to prioritize those for downlink, perhaps dropping low priority images due to bandwidth limitations.

Opportunistic Science

A third technique to improve Mars access involves opportunistically gathering science data while interacting with a poorly-modeled environment. For instance, consider the MER rovers that will travel to Mars in 2003. Each rover can travel up to 100 meters per Martian day. Given this distance, there are high probabilities that a rover will detect unexpected observation opportunities between downlinks. Capturing these opportunities involves extending the remote science processing to initiate new observations as well as prioritize the collected data from observations scheduled on the Earth.

3. MIDDLEWARE TECHNOLOGY

Motivation and State of the Art

Three approaches were introduced in the previous section to improve access to Mars: a) improving scientists' control of experiments; b) improving the quality of the downlinked raw data; and c) enabling opportunistic science. This section explores how middleware could assist with addressing these needs. The first approach, referring to Figure 1, suggests that the two-way conversation between the scientist and the instrument could be construed as creating an intelligent "channel". This channel would be capable of receiving high-level requests from the scientist, and supplying high-quality data in priority order. From the perspective of Figure 3, this involves migrating parts of all ground operations processes onto the spacecraft to assure that it can satisfy high level requests while avoiding risks to the system's safety [Estlin et al., 1999]. Issues such as security (instrument accessible only to authorized personnel for each context), could be addressed by the infrastructure. Both the first and the second approach (improved downlink data quality) suggest the need for

intelligent remote data handling, including techniques for remote science analysis and data-mining, data compression and fusion, and improved (e.g., event-driven) data handling. The third approach (opportunistic science) also needs improved leveraging of remote resources. In turn, these improvements motivate accumulation of reusable remote IT resources (such as intelligent data handling as mentioned), which can simultaneously address more than one set of needs.

Looking more closely at the applicable features of middleware itself, there are many "standards" currently in widespread use. For example, in the operating system arena, an example of a *de-jure* standard is the POSIX specification (perhaps utilized in either a spacecraft or a ground system). Conversely, an example use of a *de-facto* standard is Wind River's VxWorks for real-time ground or flight systems. Other middleware standards address communications (e.g., TCP/IP or HTTP for ground communications, and CCSDS protocols such as SLE, PROX-1, and CFDP for space communications). In the distributed computing arena, both CORBA and Java (via RMI) have become widely accepted, while in data access the ODBC/JDBC, XML and STEP standards are all in common use. For IT security, SSL, SSH, IPSec, and PKI provide standards-based access control, while X.500 and LDAP provide standard directory services. Finally, systems management has CIM, SNMP, and CMIS/CMIP standards. Many technologies are associated with such middleware standards; for example, web growth has provided many browser-based technologies for remote information access and distributed human-computer interaction; transaction technology guarantees integrity of data operations; and agent technology is evolving to provide intelligent interfaces between providers and users (both of which could be machines) of data or applications. Unfortunately, there are often several technologies and standards for addressing a particular problem, hence the middleware field is broad and somewhat incoherent even when some standards have been accepted.

Middleware-enabled Software Applications

Historically, software applications were build directly on top of the platform's operating system. A later trend towards separating "user interface" from "business logic" and "data access" fostered n-tier client/server computing, and improved modularization of code. Later, abstraction of the operating system interface and improved networking technology has made building such distributed applications easier, hence the recent growth of "standards-based" middleware. Issues such as performance, service level, and software architecture need to be addressed for particular applications, in order to ensure a proposed solution meets multiple objectives (including affordability).

If several software services are defined and built using the "shared middleware" approach, then application development can be dramatically simplified, thus enabling all the approaches to improving access to Mars defined above. Modern COTS middleware can elegantly address such issues. For example, CORBA provides a service infrastructure with pluggable components, freeing applications from having to implement such features separately. Pluggable components can also be "replaced" with alternatives performing the same function but providing additional features. For example, "remote method invocation" can be replaced by "reliable remote method invocation", perhaps without the client actually needing to know how, as long as the request is properly satisfied. Such improved capabilities can be provided to applications much more simply via standards-based common services.

An example of a CORBA service that can significantly simplify application development is the "Event Service". Use of such a service avoids the developer having to implement his own event loop (for every application), in favor of "publishing" or "subscribing" to particular types of events in a standard fashion. At the science application level, such a service could help make a remote planning decision based on subscribing to information about resource location and availability. This could dramatically reduce the 6-step cycle time described above (most of Mars Sojourner rover's useful time was spent waiting for commands from Earth). Moreover, use of asynchronous event services can make resource use even more efficient; for example, a requester can perform other useful work while waiting for the asynchronous response, handled in a standard way (e.g., via a callback).

Middleware-Provided Event Service

While this concept may appear far from reality for low-cost space missions, given today's confused middleware environment, with its plethora of rapidly-obsolete standards, there are examples of this approach being successfully prototyped in other contexts at JPL. A first example application illustrates great simplification of client software by migration to a COTS-based service architecture. Figure 4 shows a prototype re-implementation of a monitor and control information service (MCIS) deployed a few years ago in JPL's Deep Space Network (DSN) ground system. The original implementation was an entirely custom-built publish/subscribe service costing several work years to design, implement, test, and deploy. By replacing the underlying custom service with the CORBA event service (itself built upon other standard CORBA services), with a COTS implementation based on freely-available open-source in The ACE Orb (TAO) [Schmidt et. al], most of this original code could be removed, without changing the client and server API. Apart from the benefit of significantly-reduced maintenance for the JPL-written custom code, and the leverage of 20+ work-years of effort in building TAO, other applications could then use standard CORBA event service (for very different purposes). Moreover, the service itself could even be evolved independently from the application code (e.g., to provide reliable data transfer), again potentially benefiting all service users at once, and still with little or no change to application code or API.

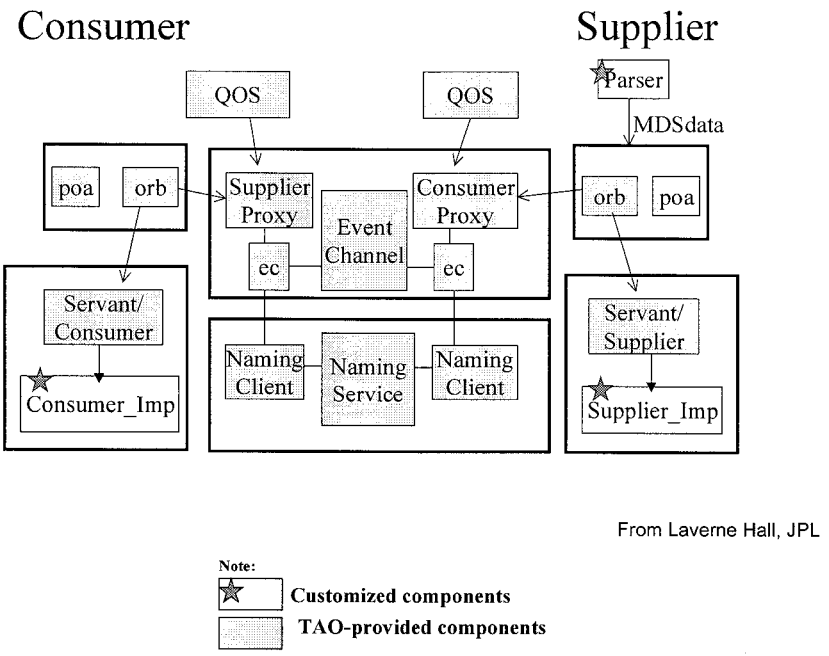

Figure 4 Example Monitor and Control Information Service Re-implementation

Middleware-Provided Data-Distribution Service

Another JPL middleware-based application currently being field tested by the US Marine Corps (shown in Figure 5), has many similarities to the information distribution architecture shown in Figure 1. Information is continuously being collected by more than 70 relatively autonomous Marine Corps units and remote sensors, geographically distributed across several disjoint areas. In order to perform their mission, the units must share time-critical information. The command center is remotely located and is responsible for mission planning, data analysis, and controlling the activities of the fielded units. It may itself be physically distributed over several miles, and is comprised of subject matter experts each interested in different facets of the data being collected and responsible for controlling different aspects of the mission. The application is responsible for ensuring that critical information

(sensor readings, commands, etc.) is shared between the components and users (both local and global) in a *timely* and *transparent* manner.

Like our example in Figure 1, the connection between the field and the command center(s) is comprised of a multi-tiered and multi-protocol network having both long-haul and local communications links. In general, a high-bandwidth communications pipe (> T1) is continuously available between nodes within the command center(s). A slightly less capable high-bandwidth communications pipe is available between a fielded unit and other units within close geographic proximity (say 2 miles). However, the long-haul communications pipes are less reliable and subject to long periods of interruption based on the loss of line-of-sight connections between the various transmitters and receivers. As a result, the long-haul pipes have an effective throughput of ~80Kbps to each node, a packet loss rate of ~30%, and packet latencies of up to 6 seconds.

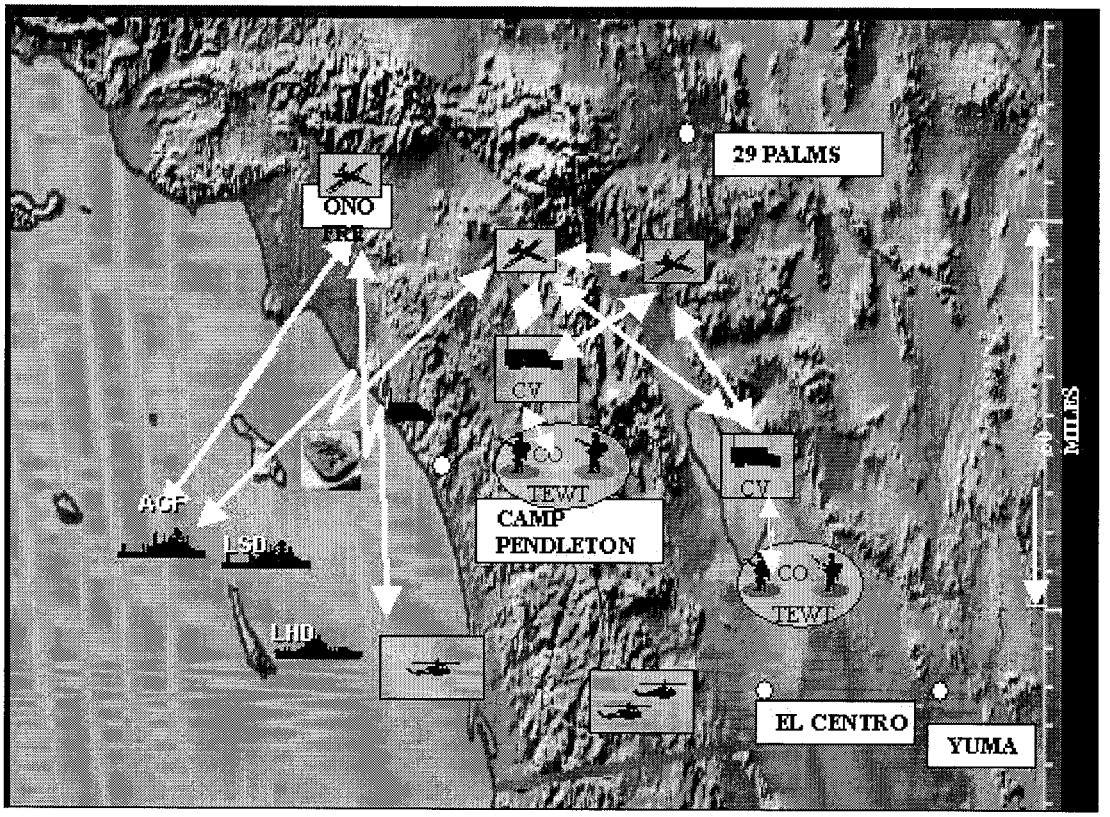

Figure 5 Three-tiered Architecture. Fielded units are connected to local communications vans (CV). Aircraft and helicopters provide long haul connections to command centers located at sea.

Application Architecture

As shown in Figure 6, the application is comprised of a eight different components written by a group of academic, government and industry collaborators using various programming languages (C++, Java, and C). The individual components execute on either UNIX servers or NT laptops located at points on the network. Depending on resource availability, components may execute on their own processor, or share process space with other components. They implement a wide variety of functions including: agent-based reasoning, situational analysis & display, data management & distribution, and interfaces to existing analysis systems and sensors. The CORBA middleware approach provided the unprecedented flexibility required for this application (different languages, different processor types, and location transparency).

The various components share information via a set of common middleware-based data-management and distribution services. They can update the shared knowledge base by making changes to the

common object repository (a.k.a. "the factory") and learn about information shared by others by querying the factory, or by subscribing to the creation, deletion or modification of specific types of information (e.g., by subscribing to changes to the mission directives assigned to me). In effect, the publish-subscribe model provides a client (or user) with an information feed tailored to their needs. The implementation of the core information management services is intentionally transparent to the clients. This approach has allowed the services to adapt to a variety of different configurations and contingencies without requiring any changes to the clients.

The core information management and distribution component is comprised of a number of middleware services:

Factory – maintains the rich object-oriented representation of the information of interest. Information is represented using a combination of real-world objects (e.g., weather, units, vehicles, mission objectives, etc.) interconnected by flexible relationships. This ontological approach allows the factory to easily maintain complex information such as the intent of individual units, or how the vehicles are collaborating to achieve a mission. The factory itself uses several additional middleware services such as: persistence, fault tolerance/replication, and association management.

Query – allows clients to retrieve objects and associations that meet specified criteria.

Publish-Subscribe (a.k.a. subscription & alerts) – allows a client to dynamically specify objects & associations of interest, and the quality of service required by the subscription. When changes are made to the factory that satisfies a subscription, the information is transparently pushed to the client on a priority and network-availability basis. The publish-subscribe mechanism is responsible for dynamically managing the flow of information across the networks in order to satisfy both static system-wide priority policies and the quality of service required by dynamic subscriptions.

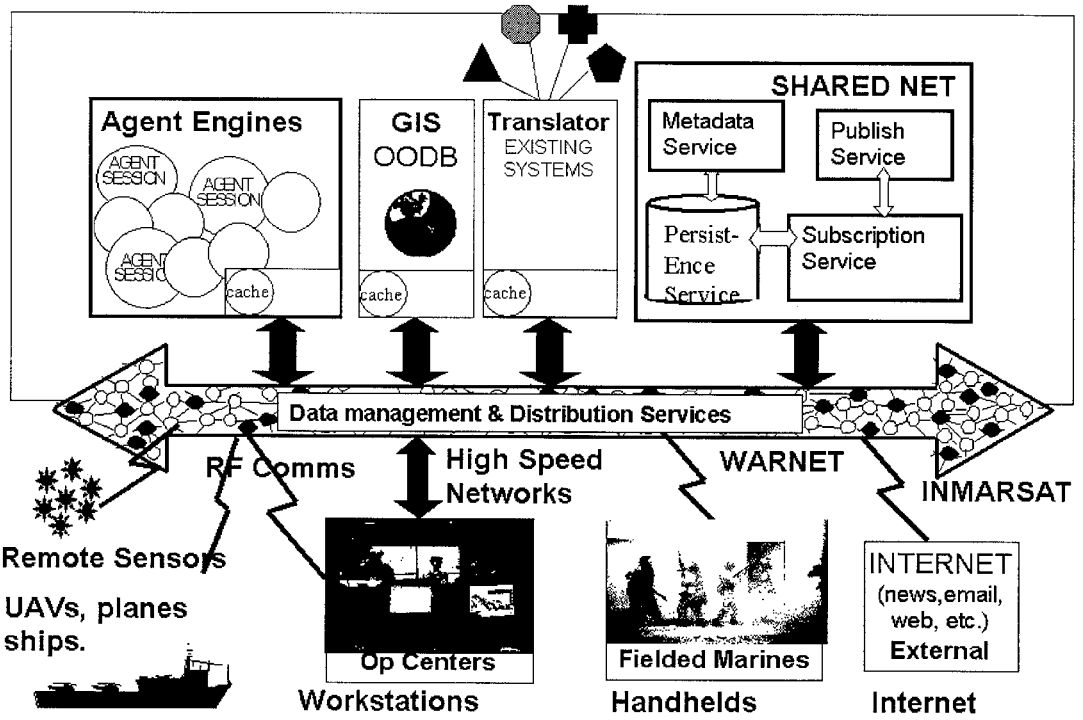

Figure 6: IMMACCS Architecture. All components communicate via a common data management and distribution service which relies heavily on middleware services.

Adaptability of the middleware approach

The middleware approach allowed the application to transparently adapt to a wide variety of unexpected situations encountered during the past year. It was originally designed to support a small group of UNIX workstations located in a command center and connected via a standard 100Mbps

Ethernet network. Near the end of the development phase, the sponsor decided to expand the scope of work to include support for up to 70 fielded Marines connected via an experimental multi-tiered RF network having significantly different characteristics than a traditional Ethernet LAN. Support for the new network not only required a complete change in the communications protocol layer (it did not support TCP/IP), but also required us to abandon the use of a single UNIX-based data management and distribution server in favor of a group of 10 cooperating NT-based servers fielded on a variety of ships, helicopters, planes, and HMMWVs spread out over an area of 100 x 200 nautical miles.

Normally, such an unforeseen requirements change would have required significant architectural redesign of both clients and servers. However, since the application was built upon middleware services, we were able to hide the changes from the clients. Components connected via a traditional Ethernet used COTS-based implementation of these services, while those connected across the experimental network used modified components specifically optimized for the experimental network. These optimizations allowed us to ensure that high-priority information was always exchanged in a timely manner, with lower priority information being exchanged as bandwidth allowed.

5. Conclusions

Both these example JPL applications demonstrate the feasibility and benefits of developing middleware-based services to improve access to Mars as described in Section 2. Under the proposed service-oriented architecture, features such as "reliable data transfer" or "fault tolerance" can be built later, using common features of the platform and infrastructure. This also provides other benefits such as improved reconfigurability for unforeseen future uses. Recently, the loss of a star tracker in the DS1 spacecraft did not result in catastrophic loss of navigation capability, because a camera was able to be reprogrammed to act as a star tracker. An on-board architecture capable of achieving this is already under development within JPL's Mission Data System [Dvorak et al., 1999]. We apply similar reasoning to the leveraging of distributed resources via middleware services. Figure 7 shows an example software architecture based on this "middleware-service approach". Some example applications are depicted, each utilizing services provided either by other sensors or resources (sensor web, rover), or by the infrastructure (data management, communication). Such an architecture also allows contemplation of an "evolvable" set of such services, progressively implemented by successive Mars missions, each contributing capability to the overall Mars environment as a secondary goal to their primary (sensor) science goals.

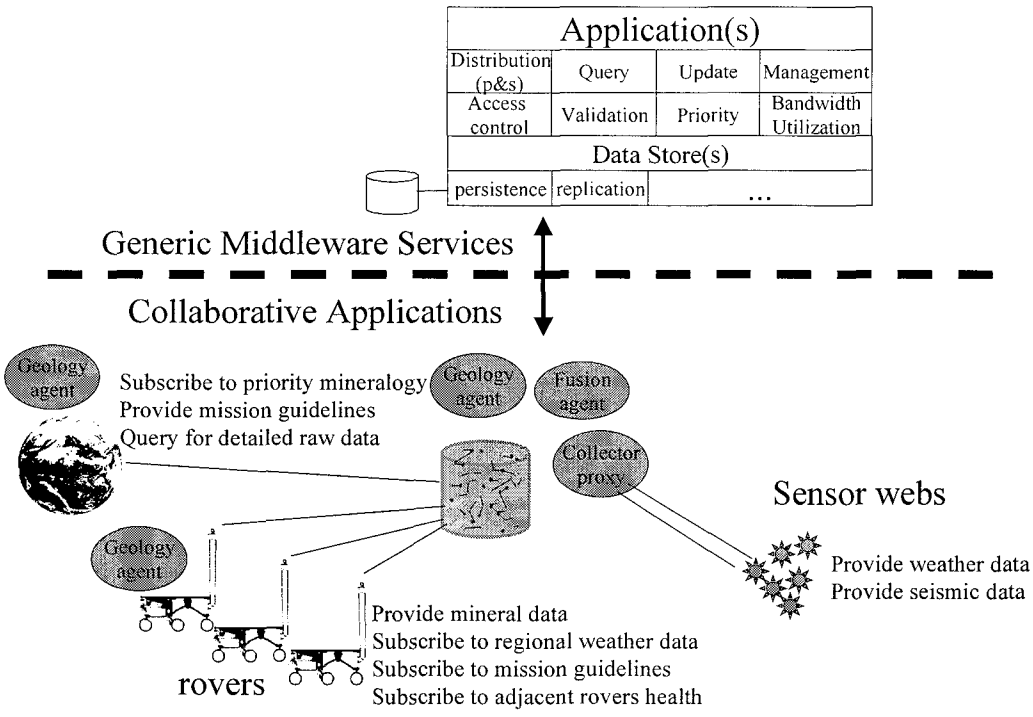

Figure 7. Example Software Architecture Enabled by Middleware Services

We believe that middleware is heading towards commodity COTS product availability. Many market segments are driving this trend, and large enterprises expect COTS middleware to solve many of their large-scale application integration problems. The consumer market demands such supplier integration; requiring coherent information access and modification hardly conceivable a decade ago. We believe that such levels of integrated "science service" are attainable for space exploration, and indeed are required in order to make such exploration more affordable (i.e., reduced failures, increased return on investment). Coupled with higher-performance communication technology (e.g., optical communication) and processor speed, we expect feasible operation of an evolvable set of middleware services on the next generation of low-cost missions. Looking further ahead, we believe the esoteric technologies of virtual multimedia and even quantum computing may be harnessed to improve the effectiveness of our interaction with remote space environments, and to progressively engage the public in such science exploration.

6. ACKNOWLEDGEMENTS

The research and design described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors wish to express appreciation for the contributions made by the MarsNetIT study team at JPL in Summer 2000, supported by JPL's Center for Space Mission Information and Software Systems (CSMISS).

7. REFERENCES

[Cesarone et al.] R. J. Cesarone, R. C. Hastrup, D. J. Bell, D. T. Lyons and K. G. Nelson, "Architectural Design for a Mars Communications & Navigation Orbital Infrastructure," presented at the AAS/AIAA Astrodynamics Specialist Conference, Girwood, Alaska, August 16-19, 1999.

[Schmidt et al.] Doug Schmidt, University of California, Irvine (http://www.cs.wustl.edu/~schmidt/)

[Mishkin, et al., 1998] A. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, B. Wilcox, "Experiences with Operations and Autonomy of the Mars Pathfinder Microrover," proceedings of the 1998 IEEE Aerospace Conference, March 21-28 1998, Snowmass at Aspen, Colorado.

[Sherwood, et al. 2000] R. Sherwood, A. Mishkin, T. Estlin, S. Chien, S. Maxwell, B. Englehardt, B. Cooper, G. Rabideau, "An Automated Rover Command Generation Prototype for the Mars 2003 Marie Curie Rover," SpaceOps 2000, Toulouse, France, June 2000.

[Manduchi, et al., 2000] R. Manduchi, S. Dolinar, A. Matache, F. Pollara, "Onboard Science Processing and Buffer Management for Intelligent Deep Space Communications," proceedings of the 2000 IEEE Aerospace Conference, March 18-25 2000, Big Sky, Montana.

[Estlin et al., 1999] T. Estlin, T. Mann, A. Gray, G. Rabideau, R. Castano, S. Chien and E. Mjolsness, "An Integrated System for Multi-Rover Scientific Exploration," Sixteenth National Conference of Artificial Intelligence (AAAI-99), July 1999, Orland, FL.

[Dvorak et al., 1999] Daniel Dvorak, Robert Rasmussen, Glenn Reeves, Al Sacks, "SOFTWARE Architecture Themes in JPL'S Mission Data System", AIAA 99-4453.

8. BIOGRAPHIES

Norman Lamarra's current primary focus is on integrating Multidisciplinary Analysis Technology for NASA's Intelligent Synthesis Environment Program, whose 5-year goal is to develop an integrated engineering environment for the conceptualization, development, and operation of future space systems. Dr. Lamarra obtained the Ph.D. degree from UCLA in System Science (Communications Systems) in 1982. Before joining JPL in 1994, he worked for over 20 years in analysis & simulation of radar systems and phased-array antennas.

Anthony Barrett is a member of the Artificial Intelligence Group, where his research and development activities involve planning and scheduling applied to controlling clusters of spacecraft and managing the operations interactions between collaborating flight projects. He holds a B.S. in Physics, Computer Science, and Applied Mathematics from James Madison University and both an M.S. and Ph.D. in Computer Science from the University of Washington. His research interests are in the areas of planning, scheduling, and multi-agent systems.